George Kolampas
Compiler Design COMP 442
Joey Paquet

29th january 2017

Assignment #1

Atomic lexical elements of the language

```
      id ::= letter alphanum*
alphanum ::= letter | digit | _
 integer ::= nonzero digit* | 0
   float ::= integer fraction [e[+|-] integer]
fraction ::= .digit* nonzero | .0
  letter ::= a..z |A..Z
   digit ::= 0..9
 nonzero ::= 1..9
```

Operators, punctuation and reserved words

```
==      +       (       if
<>      -       )       then
<       *       {       else
>       /       }       for
<=      =       [       class
>=      and     ]       int
;       not     /*      float
,       or      */      get
.               //      put
:                       return
::                      program
```

I have made no changes to the specifications given. The only thing that I must clear up is that any ambiguity between shorter sequence and long sequence the long sequence will win I.e assume ===, this will lead to == =.

The possible errors are:
line col invalid identifier
line col not an accepted character

When errors are encountered I skip till there is an acceptable lexeme and continue returning types from there.

The Token I return is:
Type string
Lexeme string
Location string – format is line col

My structure is simple, effectively I have a huge case statement for each type. The Operators, punctuation and reserved words are very simply done by peeking and seeing if they match the next few letters. The more difficult cases that needed specific attention were the atomic elements of the language. The only exposed function is next token which returns the token and the constructor of the Lscanner.

I have not used any tools. The go standard library is open source and the tokenizer they use in their compilation is available but all that would need to be added would be some lexemes specific to our purposes which didn’t seem like it would be fair.

If run on linux than I have provided the executable Compiler. To run it do
./Compiler inputfile outputfileA2CC ErrorFile
I have also provided the Compiler.exe for windows. The same commands apply.
I have run this with the tester file I provided. The I have also provided the outputs which are a2cc and errors.